第1章 绪论

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

1.1 数据结构的研究内容

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

第1章 绪论

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

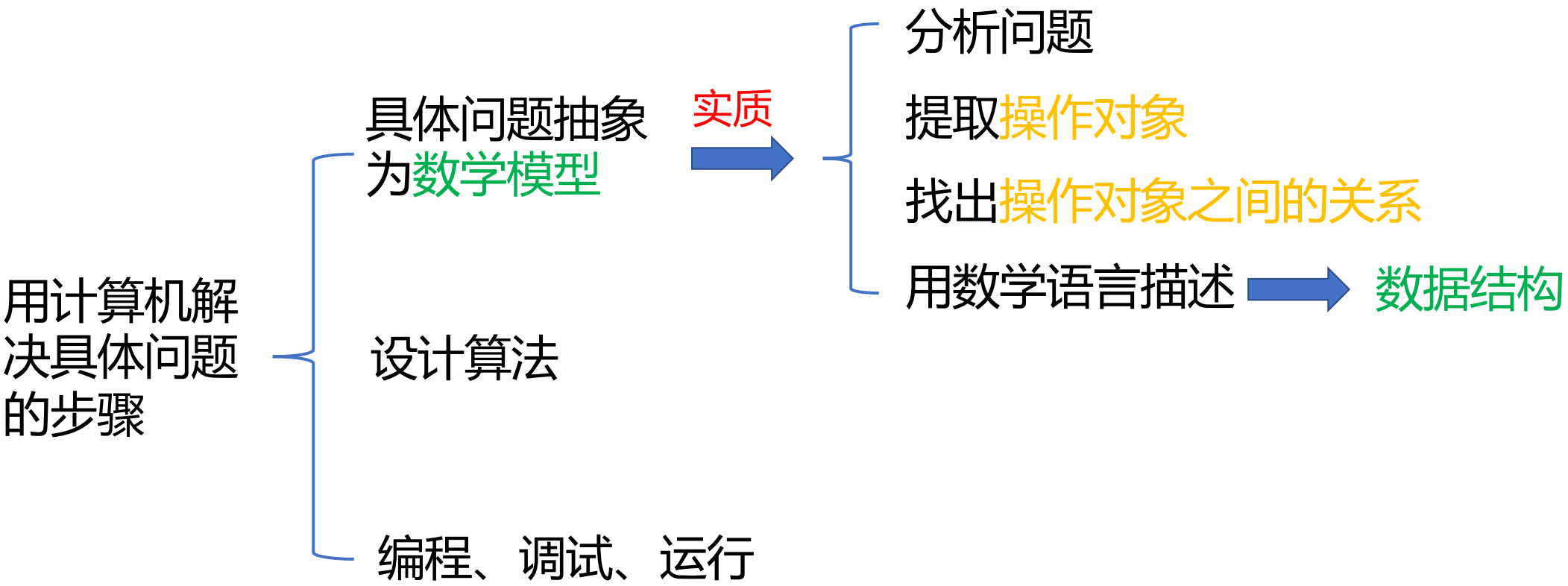
1.1 数据结构的研究内容

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

1.1 数据结构的研究内容



《数据结构》

1.1 数据结构的研究内容

- 早期，计算机主要用于数值计算
- *例1：* 求解梁架结构中的应力

  数学模型：KU=M 线性方程组

$$\begin{bmatrix} a_{11} & & \\ & \ddots & \\ & & a_{nn} \end{bmatrix} \times \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

- *例2：* 预报人口增长情况

  数学模型：微分方程

$$\begin{cases} \dfrac{dN(t)}{dt} = r\,N(t) \\ N(t)|_{t=t0} = N_0 \\ N(t) = N_0\,e^{rt} \end{cases}$$

首先，分析问题、提取操作对象；
然后，找出操作对象之间的关系，用数学语言加以描述，建立相应数学方程；
最后，求解数学方程：高斯消元法、有限元法、差分法等等

——计算数学研究范畴

特点：数据元素间的关系简单，计算复杂

1.1 数据结构的研究内容

- 随着计算机应用领域的扩展，计算机被越来越多地用于非数值计算
- *例1：* 学生学籍管理系统

学号	姓名	性别	籍贯	专业
42411	张三	男	浙江	计算机科学与技术
42412	李四	女	上海	计算机科学与技术
42413	王五	男	江苏	计算机科学与技术

线性表

操作对象：每位学生的信息（学号、姓名、性别……）

操作算法：查询、插入、修改、删除等

操作对象之间的关系：线性关系

数据结构：线性数据结构、线性表

1.1 数据结构的研究内容

- 类似的还有：图书管理系统、仓库账目管理系统、人事管理系统等

001	高等数学	樊映川	S01	…
002	理论力学	罗远祥	L01	…
003	高等数学	华罗庚	S01	…
004	线性代数	栾汝书	S02	…
⋮	⋮	⋮	⋮	⋮

高等数学	001,003,…
理论力学	002,…
线性代数	004,…
⋮	

樊映川	001,…
华罗庚	003,…
栾汝书	004,…
⋮	

L	002,…
S	001,003,…
⋮	

操作对象：若干行数据记录

操作算法：查询、插入、修改、删除等

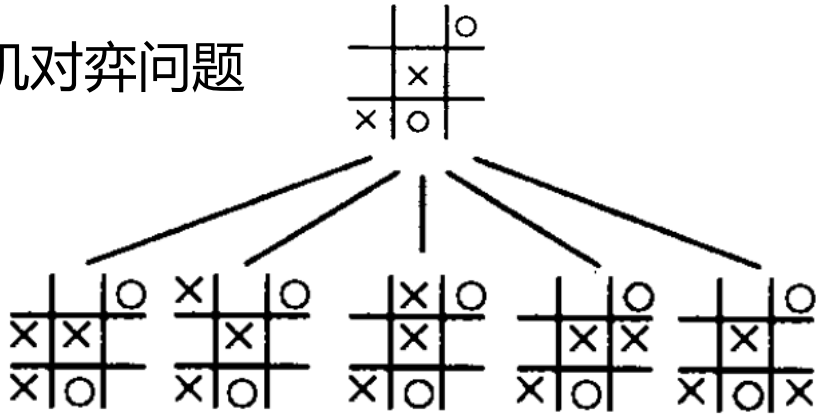
操作对象之间的关系：线性关系

数据结构：线性数据结构、线性表

1.1 数据结构的研究内容

- *例2：* 人机对弈问题

派生格局



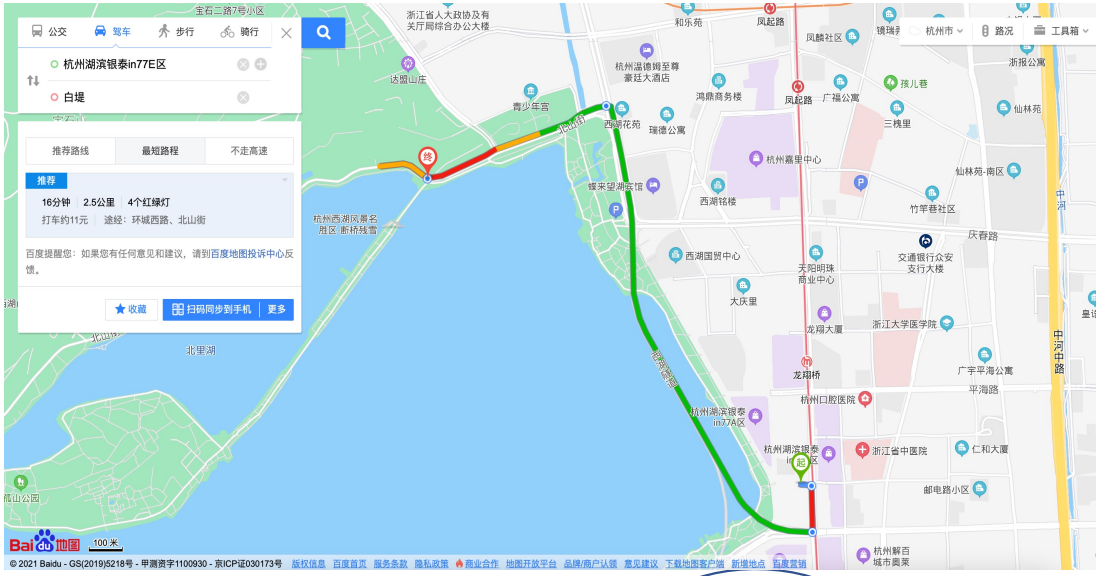
树

- 之所以能对弈：策略已经输入计算机，可以根据当前棋盘格局，来预测棋局发展的趋势，甚至最后结局。
- 操作对象：各种棋局状态，即描述棋盘的格局信息
- 操作算法：走棋，即选择一种策略使棋局状态发生变化（由一个格局派生出另一个格局）
- 操作对象之间的关系：非线性　　数据结构：树

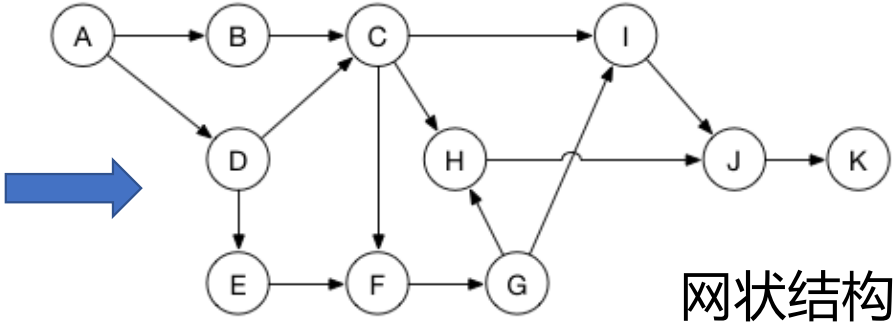
1.1 数据结构的研究内容

杭州电子科技大学 HANGZHOU DIANZI UNIVERSITY

- *例3：* 地图导航——求最短路径



图



网状结构

- 问题：找到图中两点之间最短路径
- 操作对象：各地点及路径信息
- 操作算法：选择下一个地点使路径累计长度最短
- 操作对象之间的关系：非线性关系
- 数据结构：图

数据结构研究内容小结

- 这些问题的共性是无法用数学的公式或方程来描述，是一些非数值计算的程序设计问题。
- 描述非数值计算问题的数学模型不是数学方程，而是诸如表、树、图之类的具体的数据结构。
- **数据结构**是一门研究非数值计算的程序设计问题中计算机的操作对象以及它们之间的关系和操作的学科。

要想有效地使用计算机，就必须学习数据结构

第1章 绪论

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

1.1 数据结构的研究内容

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

1.2.1 数据、数据元素、数据项和数据对象

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 数据（Data）
- 数据元素（Data Element）
- 数据项（Data Item）
- 数据对象（Data Object）

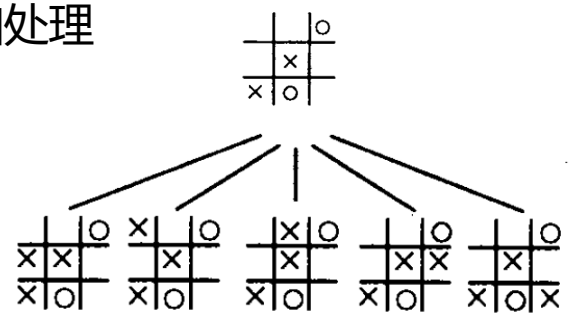
1. 数据（Data）

杭州电子科技大学 HANGZHOU DIANZI UNIVERSITY

- **数据**是能输入计算机且能被计算机程序处理的符号的总称
  - 是信息、概念与知识的载体
  - 是对客观事物的符号化表示
  - 能够被计算机识别、存储和加工
- 包括：
  - 数值型数据：整数、实数等
  - 非数值型数据：文字、图像、声音等

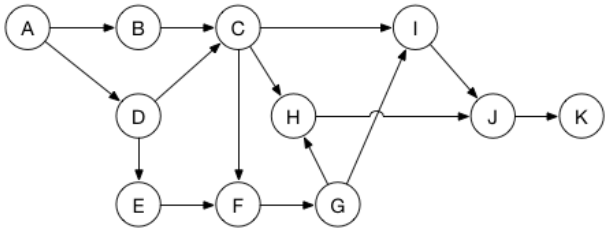
2. 数据元素（Data Element）

- 数据元素
  - 是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理
  - 也简称为元素，或称为记录、结点或顶点
  - 一个数据元素可由若干个数据项组成



学籍表				
学号	姓名	性别	籍贯	专业
42411	张三	男	浙江	计算机科学与技术
42412	李四	女	上海	计算机科学与技术
42413	王五	男	江苏	计算机科学与技术

数据元素



4. 数据项（Data Item）

- 数据项
  - 构成数据元素的不可分割的最小单位

学籍表				
学号	姓名	性别	籍贯	专业
42411	张三	男	浙江	计算机科学与技术
42412	李四	女	上海	计算机科学与技术
42413	王五	男	江苏	计算机科学与技术

- 数据、数据元素、数据项三者之间的关系
  - **数据 > 数据元素 > 数据项**

例如：学籍表 > 个人记录 > 学号、姓名……

4. 数据对象（Data Object）

- 数据对象
  - 是性质相同的数据元素的集合，是数据的一个子集

例如：

- 整数数据对象是集合N={0, ±1, ±2, ...}
- 字母字符数据对象是集合C={ ‘A’ , ‘B’ ,..., ‘Z’ }
- 学籍表也可看作一个数据对象

数据元素与数据对象

- 数据元素——组成数据的基本单元
  - 与数据的关系：是集合的个体
- 数据对象——性质相同的数据元素的集合
  - 与数据的关系：集合的子集

1.2.2 数据结构（Data Structure）

- 数据结构
  - 数据元素不是孤立存在的，他们之间存在着某种关系，数据元素相互之间的关系称为结构（Structure）
  - 是指相互之间存在一种或多种特定关系的数据元素集合
  - 或者说，数据结构是带结构的数据元素的集合

数据结构的二元组表示：
$Data\_Structure = (D, S)$
其中：D是数据元素的有限集，S是D上关系的有限集

1.2.2 数据结构（Data Structure）

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 数据结构包括以下三个方面的内容：
  - 数据元素之间的逻辑关系，也称为逻辑结构
  - 数据元素及其关系在计算机内存中的表示（映射），称为数据的物理结构或数据的存储结构
  - 数据的运算和实现，即对数据元素可以施加的操作以及这些操作在相应的存储结构上的实现

数据结构的两个层次

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

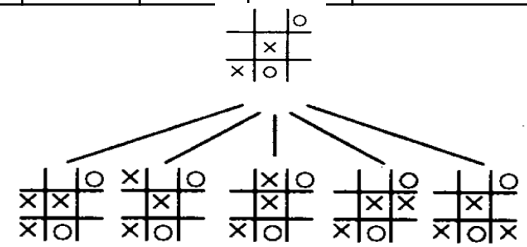
- 逻辑结构
  - 描述数据元素之间的逻辑关系
  - 与数据的存储无关，独立于计算机
  - 是从具体问题抽象出来的数学模型
- 物理结构（存储结构）
  - 数据元素及其关系在计算机存储器中的结构（存储方式）
  - 是数据结构在计算机内部的表示
- **逻辑结构与存储结构的关系：**
  - 存储结构是逻辑关系的映象与元素本身的映象
  - 逻辑结构是数据结构的抽象，存储结构是数据结构的实现
  - 两者综合起来建立了数据元素之间的结构关系

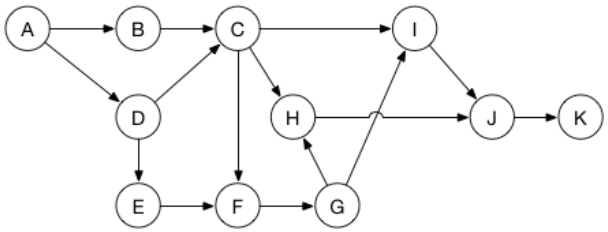
逻辑结构的种类

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

1. 线性结构
   - 有且仅有一个开始和一个终端结点，并且所有节点都最多只有一个直接前驱和一个直接后继（例如：线性表、栈、队列、串）
2. 非线性结构
   - 一个结点可能有多个直接前驱和直接后继（例如：树、图）

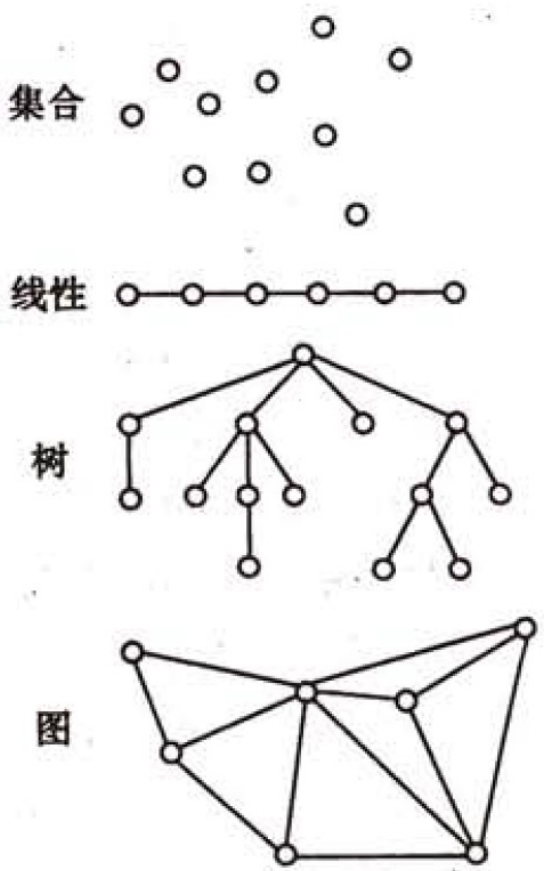
学籍表				
学号	姓名	性别	籍贯	专业
42411	张三	男	浙江	计算机科学与技术
42412	李四	女	上海	计算机科学与技术
42413	王五	男	江苏	计算机科学与技术





逻辑结构的种类

1. 集合结构
   - 结构中的数据元素之间除了同属一个集合的关系外，无任何其他关系
2. 线性结构
   - 结构中的数据元素之间存在一对一的线性关系
3. 树形结构
   - 结构中的数据元素之间存在一对多的层次关系
4. 图状结构或网状结构
   - 结构中的数据元素之间存在多对多的任意关系



存储结构的种类

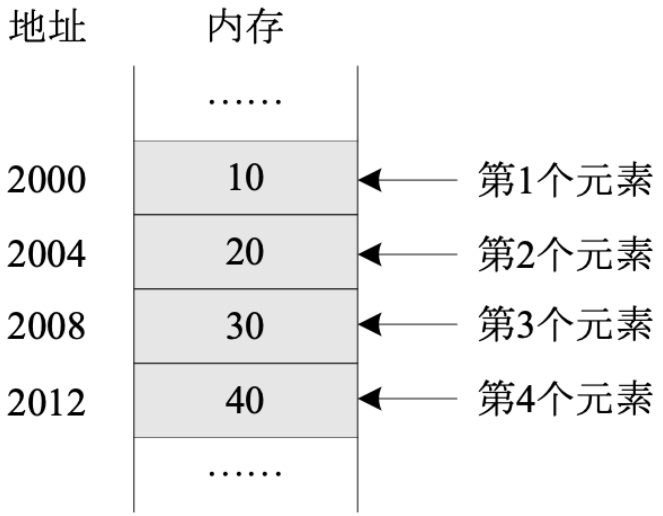
四种基本的存储结构：

1. 顺序存储结构
2. 链式存储结构
3. 索引存储结构
4. 散列存储结构

四种基本的存储结构

- 顺序存储结构
  - 用一组连续的存储单元依次存储数据元素，数据元素之间的逻辑关系由元素的存储位置来表示
  - C语言中用数组来实现顺序存储结构

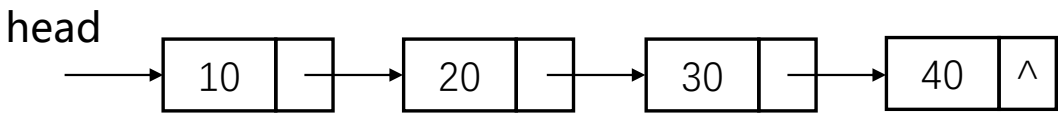
例：存储数列（10，20，30，40）

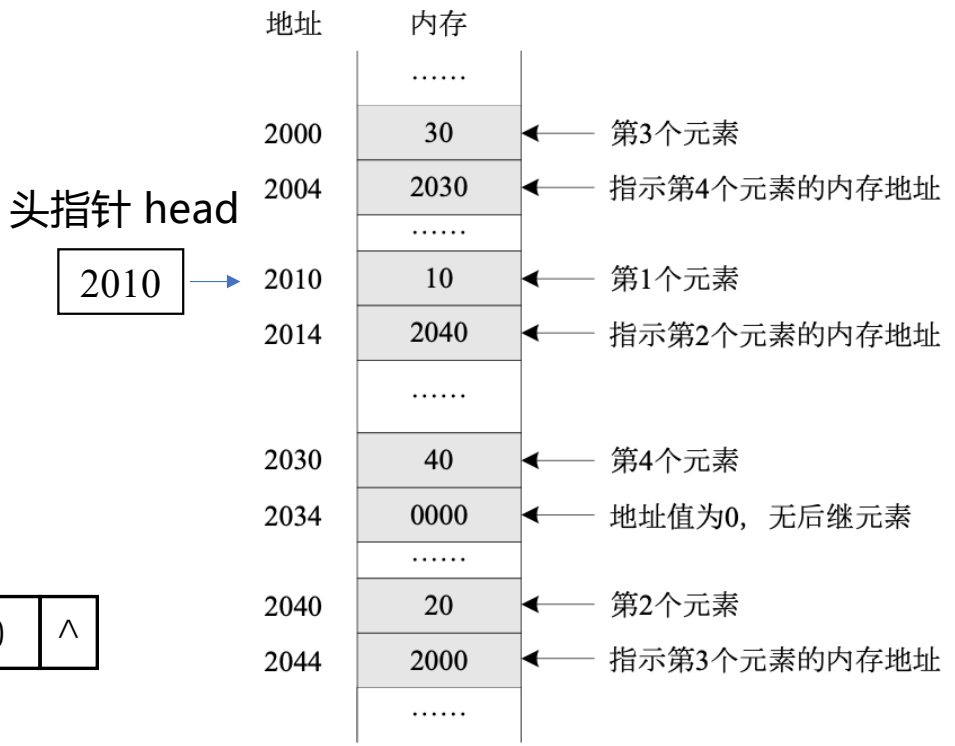


四种基本的存储结构

- 链式存储结构
  - 用一组任意的存储单元存储数据元素，数据元素之间的逻辑关系用指针来表示
  - C语言中用指针来实现链式存储结构

例：存储数列（10，20，30，40）





1.2.3 数据类型和抽象数据类型

- 在使用高级程序设计语言编写程序时，必须对程序中出现的每个变量、常量或表达式，明确说明它们所属的数据类型
  - 例如，C语言中：
    - 提供 int, char, float, double等基本数据类型
    - 数组、结构、共用体、枚举等构造数据类型
    - 指针、空（void）类型
    - 用户也可用 typedef 自定义数据类型
- 一些最基本数据结构可以用数据类型来实现，如数组、字符串等
- 而另一些常用的数据结构，如栈、队列、树、图等，不能用数据类型直接表示

1.2.3 数据类型和抽象数据类型

- 高级语言中的数据类型明显或隐含地规定了在程序执行期间变量或表达式的所有可能的取值范围，以及在这些数值范围上所允许进行的操作
  - 例如，C语言中定义变量 i 为 int 类型，就表示 i 是[min, max]范围内的整数，在这个整数集上可以进行加、减、乘、除和取模等算数运算操作
  - 数据类型的作用
    - 约束变量或常量的取值范围
    - 约束变量或常量的操作

1.2.3 数据类型和抽象数据类型

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 数据类型（Data Type）
  - 定义：数据类型是一组性质相同的值的集合以及定义于这个值集合上的一组操作的总称

数据类型=值的集合+值集合上的一组操作

1.2.3 数据类型和抽象数据类型

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 抽象数据类型（Abstract Data Type, ADT）
  - 定义：是指一个数学模型以及定义在此数学模型上的一组操作
    - 由用户定义，从问题抽象出数据模型（逻辑结构）
    - 还包括定义在数据模型上的一组抽象运算（相关操作）
    - 不考虑计算机内的具体存储结构与运算的具体实现算法

1.2.3 数据类型和抽象数据类型

- 抽象数据类型的形式定义

抽象数据类型可用（D, S, P）三元组表示

其中：D是数据对象；

S是D上的关系集；

P是对D的基本操作集

1.2.3 数据类型和抽象数据类型

- 一个抽象数据类型的定义格式：

```
ADT 抽象数据类型名{
        数据对象：<数据对象的定义>
        数据关系：<数据关系的定义>
        基本操作：<基本操作的定义>
} ADT 抽象数据类型名
```

- 其中：
  - 数据对象、数据关系的定义用伪代码描述
  - 基本操作的定义格式如下：

```
基本操作名（参数表）
        初始条件：<初始条件描述>
        操作结果：<操作结果描述>
```

1.2.3 数据类型和抽象数据类型

- 基本操作定义格式说明
  - 参数表：赋值参数 只为操作提供输入值

    引用参数 以&打头，除可提供输入值外，还将返回操作结果
  - 初始条件：描述操作执行之前数据结构和参数应满足的条件，若不满足，则操作失败，并返回相应出错信息。若初始条件为空，则省略之。
  - 操作结果：说明操作正常完成之后，数据结构的变化状况和应返回的结果。

抽象数据类型（ADT）定义举例：Circle的定义

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

ADT 抽象数据类型名{
    Data
        数据对象的定义
        数据元素之间逻辑关系的定义
    Operation
        操作1
            初始条件
            操作结果描述
        操作2
            ……
        操作2
            ……
} **ADT** 抽象数据类型名

ADT Circle{
    数据对象：D={r,x,y|r,x,y均为实数}
    数据关系：R={<r,x,y>|r是半径,<x,y>是圆心坐标}
    基本操作：
    Circle(&C, r, x, y)
      操作结果：构造一个圆。
    double Area (C)
      初始条件：圆已存在。
      操作结果：计算面积。
    double Circumference (C)
      初始条件：圆已存在。
      操作结果：计算周长。
      ……
} **ADT** Circle

抽象数据类型（ADT）定义举例：复数的定义

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

ADT 抽象数据类型名{
    Data
        数据对象的定义
        数据元素之间逻辑关系的定义
    Operation
        操作1
            初始条件
            操作结果描述
        操作2
            ……
        操作n
            ……
} **ADT** 抽象数据类型名

ADT Complex{
    数据对象：D={r1, r2|r1, r2均为实数}
    数据关系：R={<r1, r2>|r1是实部，r2是虚部}
    基本操作：
    Assign (&C, v1, v2)
      初始条件：空的复数C已存在。
      操作结果：构造复数C，r1, r2分别被赋以参数v1, v2的值。
    Destroy (&C)
      初始条件：复数C已存在。操作结果：复数C被销毁。
    GetReal(C, &realPart)
      初始条件：复数C已存在。操作结果：用realPart返回C的实部值
    GetImag(C, &imagPart)
      初始条件：复数C已存在。操作结果：用imagPart返回C的虚部值
} **ADT** Complex

第1章 绪论

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

1.1 数据结构的研究内容

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

1.3 抽象数据类型的表示与实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

- 一个问题抽象为一个抽象数据类型后，仅是形式上的抽象定义，还没有达到问题解决的目的，要实现这个目标，就要把抽象的变成具体的，即抽象数据类型在计算机上实现，变为一个能用的具体的数据类型。

ADT Complex{

数据对象：D={r1, r2|r1, r2均为实数}

数据关系：R={<r1, r2>|r1是实部，r2是虚部}

基本操作：

Assign (&C, v1, v2)

初始条件：空的复数C已存在。

操作结果：构造复数C，r1, r2分别被赋以参数v1, v2的值。

Destroy (&C)

初始条件：复数C已存在。操作结果：复数C被销毁。

GetReal(C, &realPart)

初始条件：复数C已存在。操作结果：用realPart返回C的实部值

GetImag(C, &imagPart)

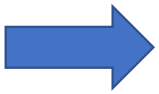
初始条件：复数C已存在。操作结果：用imagPart返回C的虚部值

} **ADT** 抽象数据类型名

抽象数据类型的实现

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

ADT 抽象数据类型名{
    Data
        数据对象的定义
        数据元素之间逻辑关系的定义
    Operation
        操作1
            初始条件
            操作结果描述
        操作2
            ……
        操作n
            ……
} **ADT** 抽象数据类型名



C语言实现抽象数据类型

- 用已有数据类型定义描述它的存储结构
- 用函数定义描述它的操作

在程序中使用

抽象数据类型如何实现

- 抽象数据类型可以通过固有的数据类型（如整型、字符型等）来表示和实现
  - 即利用处理器中已存在的数据类型来说明新的结构，用已经实现的操作来组合新的操作

注：在本门课程学习过程中，我们使用类C语言（介于伪代码和C语言之间）作为描述工具。

描述语法见教材P10-11。

用C语言真正实现抽象数据类型的定义

- 例：抽象数据类型“复数”的实现

```
Typedef struct {
        float realpart;                    /*实部*/
        float imagpart;                    /*虚部*/
} Complex                                  /*定义复数抽象类型*/

void assign (Complex *A, float real, float imag);          /*赋值*/
void add (Complex *c, Complex A, , Complex B);             /*A + B*/
void minus (Complex *c, Complex A, , Complex B);           /*A - B*/
void multiply (Complex *c, Complex A, , Complex B);        /*A * B*/
void divide (Complex *c, Complex A, , Complex B);          /*A / B*/
```

用C语言真正实现抽象数据类型的定义

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

```
void assign (Complex *A, float real, float imag){
        A->realpart = real;                          /*实部赋值*/
        A->imagpart = imag;                          /*虚部赋值*/
}                                                    /*End of assign()*/

void add (Complex *c, Complex A, , Complex B){       /*c = A + B*/
        c->realpart = A.realpart + B.realpart;       /*实部相加*/
        c->imagpart = A.imagpart + B.imagpart;      /*虚部相加*/
}                                                    /*End of add()*/
……
```

第1章 绪论

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

1.1 数据结构的研究内容

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

1.4 算法和算法分析

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

《数据结构》的研究内容

研究对象的特性及其相互之间的关系	有效地组织计算机存储	有效地实现对象之间的“运算”关系
逻辑结构	存储结构	算法

1.4 算法和算法分析

- 算法的定义
  - 对特定问题求解方法和步骤的一种描述，它是指令的有限序列。其中，每个指令表示一个或多个操作。

简而言之，算法就是解决问题的方法和步骤

Step 1: ……

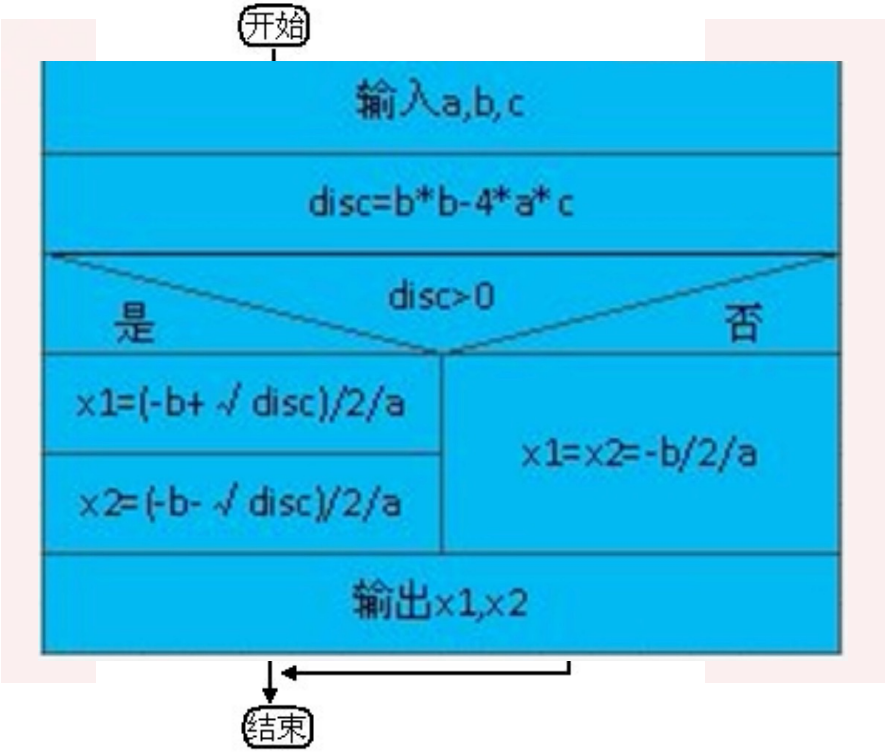
Step 2: ……

Step 3: ……

……

1.4 算法和算法分析

- 算法的描述
  - 自然语言：英语、中文
  - 流程图：传统流程图、NS流程图
  - 伪代码：类语言：类C语言
  - 程序代码：C语言程序、JAVA语言程序



1.4 算法和算法分析

- 算法与程序
  - 算法是解决问题的一种方法或一个过程，考虑如何将输入转换成输出，一个问题可以有多种算法。
  - 程序是用某种程序设计语言对算法的具体实现。

程序 = 数据结构 + 算法

- 数据结构通过算法实现操作
- 算法根据数据结构设计程序

1.4 算法和算法分析

- 算法特性：一个算法必须具备以下五个重要特性
  - 有穷性： 一个算法必须总是在执行有穷步之后结束，且每一步都可在有穷时间内完成。
  - 确定性： 算法中的每一条指令必须有确切的含义，没有二义性 。在任何条件下，算法都只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。
  - 可行性：一个算法是能行的，即算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现。
  - 输入：一个算法有零个或多个输入。
  - 输出：一个算法有一个或多个输出。

1.4 算法和算法分析

- 算法设计的要求
  - 正确性（Correctness）
  - 可读性（Readability）
  - 健壮性（Robustness）
  - 高效性（Efficiency）

正确性：算法满足问题要求，能正确解决问题

算法转化为程序后需要注意：

① 程序中不含语法错误；

② 程序对于几组输入数据能够得出满足要求的结果；

③ 程序对于精心选择的、典型、苛刻且带有刁难性的几组输入数据能够得出满足要求的结果；

④ 程序对于一切合法的输入数据都能得出满足要求的结果。

通常以第三层意义上的正确性作为衡量一个算法是否合格的标准

1.4 算法和算法分析

- 算法设计的要求
  - 正确性（Correctness）
  - 可读性（Readability）
  - 健壮性（Robustness）
  - 高效性（Efficiency）

可读性：

① 算法主要是为了人的阅读和交流，其次才是为计算机执行，因此算法应该易于人的理解；

② 另一方面，晦涩难读的算法易于隐藏较多错误而难以调试。

1.4 算法和算法分析

- 算法设计的要求
  - 正确性（Correctness）
  - 可读性（Readability）
  - 健壮性（Robustness）
  - 高效性（Efficiency）

健壮性：

① 指当输入非法数据时，算法恰当地做出反应或进行相应处理，而不是产生莫名其妙的输出结果。

② 处理出错的方法，不应是中断程序的执行，而应是返回一个表示错误或错误性质的值，以便在更高的抽象层次上进行处理。

1.4 算法和算法分析

- 算法设计的要求
  - 正确性（Correctness）
  - 可读性（Readability）
  - 健壮性（Robustness）
  - 高效性（Efficiency）

高效性：

要求花费尽量少的时间和尽量低的存储需求。

1.4 算法和算法分析

- 对于同一个问题，可以有不同的算法。如何评价这些算法的优劣？

  ——算法分析

- 算法分析的目的是看算法实际是否可行，并在同一问题存在多种算法时可进行性能上的比较，以便从中挑选出比较优的算法。

1.4 算法和算法分析

- 在满足正确性、可读性、健壮性的情况下，主要考虑算法的效率，通过算法效率的高低评判不同算法的优劣。
- 算法效率从两个方面来考虑：
  - 时间效率：算法所耗费的时间；
  - 空间效率：算法执行过程中所耗费的存储空间。

➢ 时间效率和空间效率有时候是矛盾的。

1.4 算法和算法分析

- 算法时间效率的度量
  - 算法时间效率可以用依据该算法编制的程序在计算机上执行所消耗的时间来度量
  - 两种度量方法
    - 事后统计
      - ✓ 将算法实现，测算其时间和空间开销。
      - ✓ 缺点：编写程序实现算法将花费较多时间和精力；所得实验结果依赖于计算机软硬件等环境因素，掩盖算法本身的优劣
    - 事前分析
      - ✓ 对算法所消耗资源的一种估算方法

1.4 算法和算法分析

- 事前分析方法：
  - 一个算法的运行时间是指一个算法在计算机上运行所耗费的时间，大致可以等于计算机执行一种简单的操作（如赋值、比较、移动等）所需的时间与算法中进行的简单操作次数**乘积**。

    算法运行时间 = 一个简单操作所需的时间 × 简单操作次数

  - 即算法中每条语句的执行时间之和

    算法运行时间 = ∑每条语句的执行次数 × 该语句执行一次所需的时间

    语句频度

1.4 算法和算法分析

算法运行时间 = ∑每条语句频度× 该语句执行一次所需的时间

- 每条语句执行一次所需的时间，一般随机器而异。取决于机器的指令性能、速度以及编译的代码质量。是由机器本身软硬件环境决定的，与算法无关。
- 所以，我们可假设执行每条语句所需的时间均为单位时间。此时对算法的运行时间的讨论就可转化为讨论该算法中所有语句的执行次数，即语句频度之和。
- 可以独立于不同机器的软硬件环境来分析算法的时间性能了。

1.4 算法和算法分析

- 例子：两个n × n矩阵相乘的算法：

```
1.  for (i=1; i<=n; ++i)                    //n+1次
2.      for (j=1; j<=n; ++j) {              //n(n+1)次
3.          c[i,j] = 0;                     //n*n次
4.          for (k=1; k<=n; ++k)            //n*n*(n+1)次
5.              c[i,j] += a[i,k]*b[k,j];    //n*n*n次
6.      }
```

- 算法所耗费的时间定义为该算法中每条语句的频度之和，则上述算法的时间消耗T(n)为：

一个关于n的函数：$T(n) = 2n^3 + 3n^2 + 2n + 1$

算法时间复杂度的渐进表示法

- 为了便于比较不同算法的时间效率，我们仅比较她们的数量级

  例如：两个不同的算法，时间消耗分别是：

  $T_1(n) = 10n^2$ 与 $T_2(n) = 5n^3$

  哪个好?

- 若有某个辅助函数f(n)，使得当n趋近于无穷大时，T(n)/f(n)的极限值为不等于零的常数，则称f(n)是T(n)的同数量级函数。记作T(n)=O(f(n))，称O(f(n))为算法的渐进时间复杂度，简称时间复杂度。

算法时间复杂度的渐进表示法

- 对于求解矩阵相乘问题，算法耗费时间：

$$T(n) = 2n^3 + 3n^2 + 2n + 1$$

- $n \to \infty$时，$T(n)/n^3 \to 2$，这表示n充分大时，$T(n)$与n^3是同阶或同数量级，引入大“O”记号，则$T(n)$可记作：

$$T(n) = O(n^3)$$

矩阵相乘问题的算法的渐进时间复杂度

- 一般情况下，不必计算所有操作的执行次数，只需考虑算法中基本操作执行的次数，它是问题规模n的某个函数，用T(n)表示。

算法时间复杂度定义

- 算法中基本语句重复执行的次数是问题规模n的某个函数f(n)，算法的时间度量记作： $T(n)=O(f(n))$
- 它表示随着n的增大，算法执行时间的增长率和f(n)的增长率相同，称渐进时间复杂度。

算法时间复杂度定义

- 算法中基本语句重复执行的次数是问题规模n的某个函数f(n)，算法的时间度量记作： T(n)=O(f(n))

  - 算法中重复执行次数和算法的执行时间成正比的语句
  - 对算法运行时间的贡献最大
  - 执行次数最多
  - 多数时候，是最深层循环内的语句

n越大，算法执行时间越长

- 排序：n为记录数
- 矩阵：n为矩阵阶数
- 多项式：n为多项式的项数
- 集合：n为元素个数
- 树：n为树的结点个数
- 图：n为图的顶点数或边数

分析算法时间复杂度的基本方法

- 定理：

若$f(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n^1 + a_0$是k次多项式，则$T(n) = O(n^k)$。

忽略所有低次幂项和最高次幂项系数，体现出增长率的含义

分析算法时间复杂度的基本方法

① 找出语句频度最大的那条语句作为基本语句

② 计算基本语句的频度得到问题规模n的某个函数f(n)

③ 取其数量级用符号“O”表示

```
for ( int k = 0; k < n; k ++ )
    x ++;
for ( int i = 0; i < n; i++ )
   for ( int j = 0; j < n; j++ )
      y ++;
```

$T(n) = n^2$

$f(n) = n(n+1)$

最深层循环内的语句

算法时间复杂度分析


```
void exam ( float x[ ][ ], int m, int n ) {
    float sum [ ];
    for ( int i = 0; i < m; i++ ) {
        sum[i] = 0.0;
        for ( int j = 0; j < n; j++ )
            sum[i] += x[i][j];
    }
```

$T(n) = O(m * n)$

$f(n) = m * n$

时间复杂度是由嵌套最深层语句的频度决定的

算法时间复杂度分析例题

- 例1：两个n × n矩阵相乘的算法：

```
for (i=1; i<=n; ++i)
    for (j=1; j<=n; ++j) {
      c[i,j] = 0;
      for (k=1; k<=n; ++k)
        c[i,j] += a[i,k]*b[k,j];
    }
```

基本操作语句

$$T(n) = O(n^3)$$

$$T(n) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} 1 = \sum_{i=1}^{n}\sum_{j=1}^{n} n = \sum_{i=1}^{n} n^2 = n^3 = O(n^3)$$

算法时间复杂度分析例题

- 例2：

```
for (i=1; i<=n; i++)
    for (j=1; j<=i; j++)
        for (k=1; k<=j; k++)
            x=x+1;
```

基本操作语句

$T(n) = O(n^3)$

$$T(n)=\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j}1=\sum_{i=1}^{n}\sum_{j=1}^{i}j=\sum_{i=1}^{n}\frac{i(i+1)}{2}=\left(\sum_{i=1}^{n}\frac{i^2}{2}+\sum_{i=1}^{n}\frac{i}{2}\right)$$

$$=\frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6}+\frac{n(n+1)}{2}\right)=\frac{n(n+1)(n+2)}{6}$$

算法时间复杂度分析例题

- 例3：

```
i=1;
while(i<=n)
    i=i*2;
```

基本操作语句

$$f(n) = \log_2 n$$

$$T(n) = O(\log_2 n)$$

关键：找出执行次数x与n的关系，并表示成n的函数

若循环执行1次：i=1*2=2,

若循环执行2次：i=2*2=2^2,

若循环执行3次：i=2^2*2=2^3,

……

若循环执行x次：i=2^x

设基础操作语句执行次数为x，由循环条件i<=n 可得：$2^x <= n$, $x <= \log_2 n$

算法时间复杂度计算

- 注意：有的情况下，算法中基本操作重复执行的次数还随问题的输入数据集不同而不同

【例】 顺序查找，在数组a中查找值等于e的元素，返回其所在位置

```
for (i=0; i<n; i++)
    if (a[i]==e) return i+1;          //找到，则返回时第几个元素
return 0;
```

- 最好情况：1次
- 最坏情况：n次
- 平均时间复杂度为：O(n)

算法时间复杂度

杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

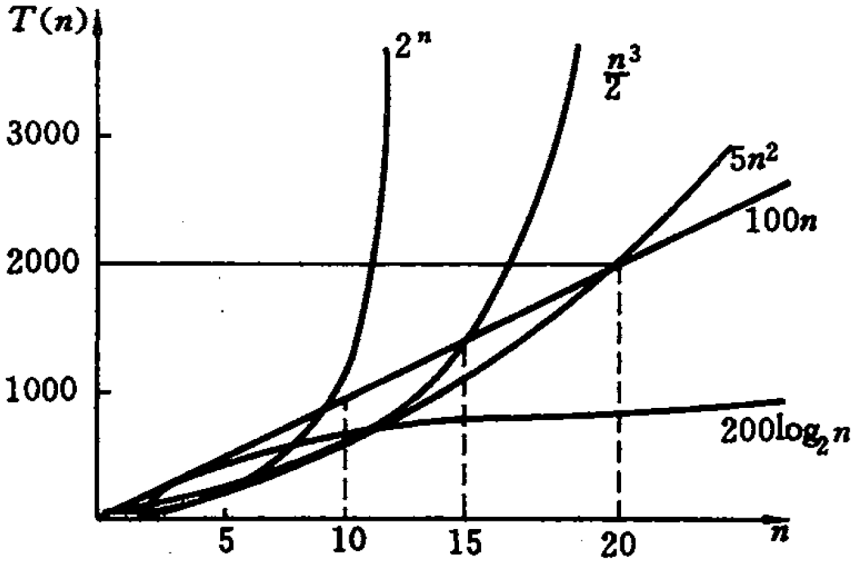
- 最坏时间复杂度：在最坏情况下，算法的时间复杂度
- 平均时间复杂度：所有可能输入实例在等概率出现的情况下，算法的期望运行时间
- 最好时间复杂度：在最好情况下，算法的时间复杂度

一般总是考虑最坏时间复杂度，以保证算法的运行时间不会比它长

算法时间效率的比较

- 当n取的很大时，指数时间算法和多项式时间算法在所需时间上非常悬殊

n	f(1)	f(logn)	f(n)	f(nlogn)	$f(n^2)$	$f(n^3)$
1	1	0	1	0	1	1
2	1	1	2	2	4	8
4	1	2	4	8	16	64
8	1	3	8	24	64	512
16	1	4	16	64	256	4096
32	1	5	32	160	1024	32768



复杂度低 → 复杂度高

常数阶	对数阶	线性阶	线性对数阶	平方阶	立方阶	...	K次方阶	指数阶
O(1)	$O(\log_2 n)$	O(n)	$O(n\log_2 n)$	$O(n^2)$	$O(n^3)$	...	$O(n^K)$	$O(2^n)$

渐进空间复杂度

- 空间复杂度：算法所需存储空间的度量，记作

$$S(n)=O(f(n))$$

其中n为问题的规模

- 算法要占据的空间
  - 算法本身要占据的空间，输入/输出，指令，常数，变量等
  - 算法要使用的辅助空间

算法空间复杂度分析例题

- 【例】将一维数组a中的n个数逆序存放到原数组中

【算法1】

```
for (i=0; i<n; i++)
    b[i]=a[n-i-1];
for (i=0; i<n; i++)
    a[i]=b[i];
}
```

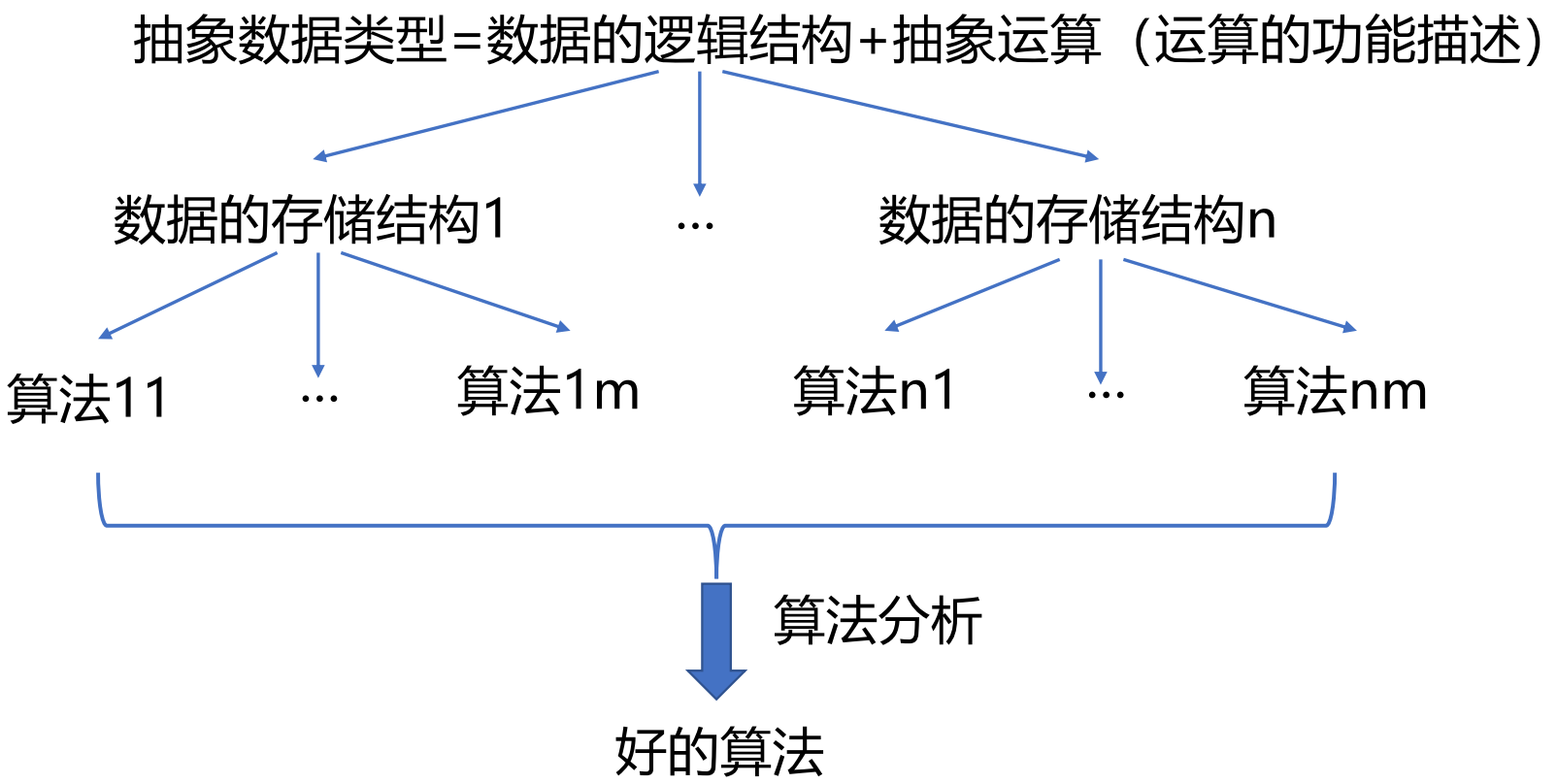
$S(n) = O(n)$

【算法2】

```
for (i=0; i<n/2; i++){
    t=a[i];
    a[i]=a[n-i-1];
    a[n-i-1]=t;
}
```

$S(n) = O(1)$ 原地工作

设计算法的过程


抽象数据类型=数据的逻辑结构+抽象运算（运算的功能描述）
数据的存储结构1
…
数据的存储结构n
算法11
…
算法1m
算法n1
…
算法nm
算法分析
好的算法